


[Subscribe \(Full Service\)](#) [Register \(Limited Service, Free\)](#) [Login](#)

 Search: ☒ The ACM Digital Library ☐ The Guide


 Searching within The ACM Digital Library for: "call stack" and "return address" ([start a new search](#))

Found 61 of 241,429

REFINER YOUR SEARCH

[Search Results](#)
[Related Journals](#)
[Related Magazines](#)
[Related SIGs](#)
[Related Conferences](#)

Results 1 - 20 of 61

 Sort by in
[Save results to a Binder](#)

 Result page: 1 2 3 4 [next](#) >

▼ Refine by Keywords

[Discovered Terms](#)

▼ Refine by People

[Names](#)

[Institutions](#)

[Authors](#)

[Reviewers](#)

▼ Refine by Publications

[Publication Year](#)

[Publication Names](#)

[ACM Publications](#)

[All Publications](#)

[Content Formats](#)

[Publishers](#)

▼ Refine by Conferences

[Sponsors](#)

[Events](#)

[Proceeding Series](#)

- [A framework for diversifying windows native APIs to tolerate code injection attacks](#)
 Lynette Qu Nguyen, Tufan Demir, Jeff Rowe, Francis Hsu, Karl Levitt
 March 2007 ASI ACCS '07: Proceedings of the 2nd ACM symposium on Information, computer and communications security

Publisher: ACM

 Full text available: [Pdf](#) (170.52 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

Bibliometrics: Downloads (6 Weeks): 6, Downloads (12 Months): 94, Citation Count: 0

We present a framework to prevent code injection attacks in MS Windows using Native APIs in the operating system. By adopting the idea of diversity, this approach is implemented in a two-tier framework. The first tier permutes the Native API dispatch .

Keywords: code injection attacks, diversity, windows native API

- [A type system for object initialization in the Java bytecode language](#)
 Stephen N. Freund, John C. Mitchell
 November 1999 Transactions on Programming Languages and Systems (TOPLAS) , Volume 21 Issue 6

Publisher: ACM

 Full text available: [Pdf](#) (394.88 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [cited by](#), [index terms](#)

Bibliometrics: Downloads (6 Weeks): 2, Downloads (12 Months): 43, Citation Count: 19

In the standard Java implementation, a Java language program is compiled to Java bytecode. This bytecode may be sent across the network to another site, where it is then executed by the Java Virtual Machine. Since bytecode may be written by hand, or ...

Keywords: Java, bytecode languages, object initialization, type checking

- [Standard fixpoint iteration for Java bytecode verification](#)
 Zhenyu Qian
 July 2000 Transactions on Programming Languages and Systems (TOPLAS) , Volume 22 Issue 4

Publisher: ACM

 Full text available: [Pdf](#) (439.82 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [cited by](#), [index terms](#)

Bibliometrics: Downloads (6 Weeks): 4, Downloads (12 Months): 37, Citation Count: 15

Java bytecode verification forms the basis for Java-based Internet security and needs a rigorous description. One important aspect of bytecode verification is to check if a Java Virtual Machine (JVM) program is statically well-typed. So far, several ...

ADVANCED SEARCH

[Advanced Search](#)


FEEDBACK

[Please provide us with feedback](#)

Found 61 of 241,429


Keywords: Java, bytecode verification, dataflow analysis, fixpoint

4 [Mondrix: memory isolation for linux using mondriaan memory protection](#)

 Emmett Witchel, Junghwan Rhee, Krste Asanović

October 2005 SOSP '05: Proceedings of the twentieth ACM symposium on Operating systems principles

Publisher: ACM

Full text available:  Pdf (332.09 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [cited by](#), [index terms](#)

Bibliometrics: Downloads (6 Weeks): 21, Downloads (12 Months): 124, Citation Count: 10


This paper presents the design and an evaluation of Mondrix, a version of the Linux kernel with Mondriaan Memory Protection (MMP). MMP is a combination of hardware and software that provides efficient fine-grained memory protection between multiple protection ...

Keywords: fine-grained memory protection

Also published in:

October 2005 SIGOPS Operating Systems Review Volume 39 Issue 5

5 [SPIKE: engineering malware analysis tools using unobtrusive binary-instrumentation](#)

 Amit Vasudevan, Ramesh Yerraballi

January 2006 ACSC '06: Proceedings of the 29th Australasian Computer Science Conference - Volume 48 , Volume 48

Publisher: Australian Computer Society, Inc.


Full text available:  Pdf (832.66 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

Bibliometrics: Downloads (6 Weeks): 4, Downloads (12 Months): 91, Citation Count: 0

Malware -- a generic term that encompasses viruses, trojans, spywares and other intrusive code -- is widespread today. Malware analysis is a multi-step process providing insight into malware structure and functionality, facilitating the development of ...

Keywords: instrumentation, malware, security

6 [Positional adaptation of processors: application to energy reduction](#)

 Michael C. Huang, Jose Renau, Josep Torrellas

June 2003 ISCA '03: Proceedings of the 30th annual international symposium on Computer architecture

Publisher: ACM

Full text available:  Pdf (225.57 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [cited by](#)


Bibliometrics: Downloads (6 Weeks): 3, Downloads (12 Months): 39, Citation Count: 36

Although adaptive processors can exploit application variability to improve performance or save energy, effectively managing their adaptivity is challenging. To address this problem, we introduce a new approach to adaptivity: the *Positional* approach. ...

Also published in:


May 2003 SIGARCH Computer Architecture News Volume 31 Issue 2

7 [Memsherlock: an automated debugger for unknown memory corruption vulnerabilities](#)

 Emre C. Sezer, Peng Ning, Chongkyung Kil, Jun Xu

October 2007 CCS '07: Proceedings of the 14th ACM conference on Computer and communications security

Publisher: ACM

Full text available:  Pdf (380.79 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

Bibliometrics: Downloads (6 Weeks): 13, Downloads (12 Months): 217, Citation Count: 0


Software vulnerabilities have been the main contributing factor to the Internet security problems such as fast spreading worms. Among these software vulnerabilities, memory corruption vulnerabilities such as buffer overflow and format string bugs have ...

Keywords: debugging, memory corruption, vulnerability analysis

8 [Eliminating stack overflow by abstract interpretation](#)

 [John Regehr](#), [Alastair Reid](#), [Kirk Webb](#)

November 2005 Transactions on Embedded Computing Systems (TECS) , Volume 4 Issue 1
Publisher: ACM


Full text available:  Pdf (510.78 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [cited by](#), [index terms](#)

Bibliometrics: Downloads (6 Weeks): 14, Downloads (12 Months): 106, Citation Count: 5

An important correctness criterion for software running on embedded microcontrollers is *stack safety*: a guarantee that the call stack does not overflow. Our first contribution is a method for statically guaranteeing stack safety of interrupt-driven ...


Keywords: Microcontroller, abstract interpretation, call stack, context sensitive, dataflow analysis, interrupt-driven, sensor network

9 [A practical mimicry attack against powerful system-call monitors](#)

 [Chetan Parampalli](#), [R. Sekar](#), [Rob Johnson](#)

March 2008 ASI ACCS '08: Proceedings of the 2008 ACM symposium on Information, computer and communications security

Publisher: ACM


Full text available:  Pdf (325.91 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

Bibliometrics: Downloads (6 Weeks): 17, Downloads (12 Months): 131, Citation Count: 0

System-call monitoring has become the basis for many host-based intrusion detection systems as well as policy enforcement techniques. Mimicry attacks attempt to evade system-call monitoring IDS by executing innocuous-looking sequences of system calls that accomplish ...

Keywords: buffer overflow, intrusion-detection, memory error, mimicry attack, system call monitor

10 [When good instructions go bad: generalizing return-oriented programming to RISC](#)

 [Erik Buchanan](#), [Ryan Roemer](#), [Hovav Shacham](#), [Stefan Savage](#)

October 2008 CCS '08: Proceedings of the 15th ACM conference on Computer and communications security

Publisher: ACM


Full text available:  Pdf (415.17 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

Bibliometrics: Downloads (6 Weeks): 31, Downloads (12 Months): 92, Citation Count: 0

This paper reconsiders the threat posed by Shacham's "return-oriented programming" a technique by which W-xor-X-style hardware protections are evaded via carefully crafted stack frames that divert control flow into the middle of existing variable-length


Keywords: RISC, SPARC, return-into-libc, return-oriented programming

11 [Speculative return address stack management revisited](#)

 [Hans Vandierendonck, André Seznec](#)

November 2008 Transactions on Architecture and Code Optimization (TACO) , Volume Issue 3

Publisher: ACM


Full text available:  [Pdf](#) (285.87 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

Bibliometrics: Downloads (6 Weeks): 41, Downloads (12 Months): 74, Citation Count: 0

Branch prediction feeds a speculative execution processor core with instructions. Branch mispredictions are inevitable and have negative effects on performance and energy consumption. With the advent of highly accurate conditional branch predictors, ...

Keywords: Return address prediction, back-up predictor, corruption detection

12 [Low-overhead call path profiling of unmodified, optimized code](#)

 [Nathan Froyd, John Mellor-Crummey, Rob Fowler](#)

June 2005 ICS '05: Proceedings of the 19th annual international conference on Supercomputing

Publisher: ACM

Full text available:  [Pdf](#) (399.57 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [cited by](#)

Bibliometrics: Downloads (6 Weeks): 10, Downloads (12 Months): 113, Citation Count: 7

Call path profiling associates resource consumption with the calling context in which resources were consumed. We describe the design and implementation of a low-overhead call path profiler based on stack sampling. The profiler uses a novel sample-driven ...

13 [Enabling Java mobile computing on the IBM Jikes research virtual machine](#)

 [Giacomo Cabri, Letizia Leonardi, Raffaele Quitadamo](#)

August 2006 PPPJ '06: Proceedings of the 4th international symposium on Principles and practice of programming in Java

Publisher: ACM


Full text available:  [Pdf](#) (389.80 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

Bibliometrics: Downloads (6 Weeks): 15, Downloads (12 Months): 44, Citation Count: 0

Today's complex applications must face the distribution of data and code among different network nodes. Java is a wide-spread language that allows developers to build complex software, even distributed, but it cannot handle the migration of computations ...

Keywords: Java virtual machine, code mobility, distributed applications, thread persistence

14 [Reducing runtime complexity of long-running application services via dynamic profiling and dynamic bytecode adaptation for improved quality of service](#)

 [John Bergin, Liam Murphy](#)

November 2007 WRASQ '07: Proceedings of the 2007 workshop on Automating service quality: Held at the International Conference on Automated Software Engineering (ASE)

Publisher: ACM


Full text available:  [Pdf](#) (294.70 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

Bibliometrics: Downloads (6 Weeks): 2, Downloads (12 Months): 55, Citation Count: 0

We present a transparent optimisation framework for automatically improving run-time performance of component-based enterprise applications. Run-time performance is improved by automatically identifying and dynamically switching to an optimised but functionally ...


Keyw ords: adaptation, caching, java language, object caching, optimisation, profiling

15 [Using DISE to protect return addresses from attack](#)

 [Marc L. Corliss](#), [E. Christopher Lewis](#), [Amir Roth](#)

March 2005 SIGARCH Computer Architecture News , Volume 33 Issue 1


Publisher: ACM

Full text available:  [Pdf](#) (389.57 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

Bibliometrics: Downloads (6 Weeks): 3, Downloads (12 Months): 40, Citation Count: 1


Stack-smashing by buffer overflow is a common tactic used by viruses and worms to crash or hijack systems. Exploiting a bounds-unchecked copy into a stack buffer, an attacker can---by supplying a specially-crafted and unexpectedly long input---overwrite ...

16 [Code quality tools: learning from our experience](#)

 [R Krishnan](#), [S Murali Krishna](#), [Nishil Bharill](#)

July 2007 SIGSOFT Software Engineering Notes , Volume 32 Issue 4

Publisher: ACM


Full text available:  [Pdf](#) (296.61 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

Bibliometrics: Downloads (6 Weeks): 8, Downloads (12 Months): 121, Citation Count: 0

In this paper we share some of our experiences relating to tools used in coding phase. We primarily focus our discussion on two topics, namely UT (Unit Testing) and Memory related errors. Unit Testing (UT) [1] is a critical early-phase verification activity ...


Keyw ords: buffer overflow and memory corruption, code quality, memory leak

17 [Design and evaluation of dynamic optimizations for a Java just-in-time compiler](#)

 [Toshio Suganuma](#), [Toshiaki Yasue](#), [Motohiro Kawahito](#), [Hideaki Komatsu](#), [Toshio Nakatani](#)

July 2005 Transactions on Programming Languages and Systems (TOPLAS) , Volume 27 Issue 4

Publisher: ACM


Full text available:  [Pdf](#) (1.60 MB) Additional Information: [full citation](#), [abstract](#), [references](#), [cited by](#), [index terms](#)

Bibliometrics: Downloads (6 Weeks): 24, Downloads (12 Months): 249, Citation Count: 6

The high performance implementation of Java Virtual Machines (JVM) and Just-In-Time (JIT) compilers is directed toward employing a dynamic compilation system on the basis of online runtime profile information. The trade-off between the compilation overhead


Keyw ords: JIT compiler, Recompilation, adaptive optimization, code specialization, dynamic compilation, profile-directed method inlining

18 [A systematic study of functional language implementations](#)

 [Rémi Douence](#), [Pascal Fradet](#)

March 1998 Transactions on Programming Languages and Systems (TOPLAS) , Volume 20 Issue 2

Publisher: ACM

Full text available:  [Pdf](#) (273.98 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [cited by](#), [index terms](#)

Bibliometrics: Downloads (6 Weeks): 4, Downloads (12 Months): 79, Citation Count: 4

We introduce a unified framework to describe, relate, compare, and classify functional language implementations. The compilation process is expressed as a succession of program transformations in the common framework. At each step, different transformations ...

Keywords: abstract machines, combinators, compilers, functional programming, program transformation

19 [A type system for Java bytecode subroutines](#)



Raymie Stata, Martin Abadi

January 1999 Transactions on Programming Languages and Systems (TOPLAS) ,
Volume 21 Issue 1

Publisher: ACM

Full text available: Pdf (519.84 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [cited by](#), [index terms](#)

Bibliometrics: Downloads (6 Weeks): 1, Downloads (12 Months): 39, Citation Count: 23

Java is typically compiled into an intermediate language, JVMIL, that is interpreted by the Java Virtual Machine. Because mobile JVMIL code is not always trusted, a bytecode verifier enforces static constraints that prevent various dynamic errors. Given ...

Keywords: Java, bytecode verification

20 [Formal certification of a compiler back-end or: programming a compiler with a proof assistant](#)



Xavier Leroy

January 2006 POPL '06: Conference record of the 33rd ACM SIGPLAN-SIGACT symposium on Principles of programming languages

Publisher: ACM

Full text available: Pdf (187.24 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [cited by](#), [index terms](#)

Bibliometrics: Downloads (6 Weeks): 23, Downloads (12 Months): 139, Citation Count: 25

This paper reports on the development and formal certification (proof of semantic preservation) of a compiler from Cminor (a C-like imperative language) to PowerPC assembly code, using the Coq proof assistant both for programming the compiler and for ...

Keywords: certified compilation, compiler transformations and optimizations, program proof, semantic preservation, the Coq theorem prover

Also published in:

January 2006 SIGPLAN Notices Volume 41 Issue 1

Result page: 1 2 3 4 next >

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2009 ACM, Inc.

[Terms of Usage](#) [Privacy Policy](#) [Code of Ethics](#) [Contact Us](#)

Useful downloads: [Adobe Acrobat](#) [QuickTime](#) [Windows Media Player](#) [Real Player](#)